Faculty of Engineering and Technology

Electrical and Computer Engineering Department

ENCS5141—Intelligent Systems Laboratory

Assignment #1: Data Cleaning and Feature Engineering & Comparative Analysis of Classification Techniques, comparing Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP)

Prepared by: Ibaa Taleeb **ID** : 1203073

Instructor : Dr.Aziz Qaroush

Section: 2

Date: 25/11/2024

Abstract

This case study investigates the use of data cleaning, feature engineering, and a comparative analysis of classification techniques on the Bike Sharing dataset. In Part 1, the focus is on preprocessing the data, which includes handling missing values, encoding categorical variables, scaling numerical features, and applying dimensionality reduction. The effect of these preprocessing steps on the performance of a Random Forest model is evaluated by comparing its accuracy, precision, and recall against a model trained on raw, unprocessed data.

In Part 2, a comparative analysis is conducted between three classification models Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) to evaluate their effectiveness in predicting bike demand. The models are assessed using performance metrics such as accuracy, precision, recall, and F-Measure, while also considering the impact of parameter tuning and preprocessing. The results provide a deeper understanding of the strengths and weaknesses of each model and their ability to predict bike demand. The assignment concludes by identifying the model that offers the best balance between prediction accuracy and computational efficiency.

Table of Contents

Table of Figures

2. Procedure and Discussion

Part 1: Data Cleaning and Feature Engineering for the Bike Sharing Dataset

1.1 Data Exploration: Summarize dataset statistics to understand the structure, features, and any missing values.

The study starts by get an overview of the dataset by displaying statistics, identifying missing values, and visualizing the relationships between numerical features, the dataset includes both numerical and categorical data, with 70,000 entries spread across 34 features.

	Insulin Levels	Age	BMI	Blood Pressure	\		Weight Gain During Pregnancy	Pancreatic Health	Pulmonary Function
count	70000.000000	70000.000000	70000.000000	70000.000000		count	70000.000000	70000.000000	70000.000000
mean	21.607443	32.020700	24.782943	111.339543		mean	15.496414	47.564243	70.264671
std	10.785852	21.043173	6.014236	19.945000		std	9.633096	19.984683	11.965600
min	5.000000	0.000000	12.000000	60.000000		min	0.000000	10.000000	30.000000
25%	13.000000	14.000000	20.000000	99.000000		25%	7.000000	32.000000	63.000000
50%	19.000000	31.000000	25.000000	113.000000		50%	16.000000	46.000000	72.000000
75%	28.000000	49.000000	29.000000	125.000000		75%	22.000000	64.000000	79.000000
max	49.000000	79.000000	39.000000	149.000000		max	39.000000	99.000000	89.000000

	Cholesterol Levels	Waist Circumference	Blood Glucose Levels		Neurological Assessments	Digestive Enzyme Levels	Birth Weight
count	70000.000000	70000.000000	70000.000000	count	70000.000000	70000.000000	70000.000000
mean	194.867200	35.051657	160.701657	mean	1.804157	46.420529	3097.061071
std	44.532466	6.803461	48.165547	std	0.680154	19.391089	713.837300
min	100.000000	20.000000	80.000000	min	1.000000	10.000000	1500.000000
25%	163.000000	30.000000	121.000000	25%	1.000000	31.000000	2629.000000
50%	191.000000	34.000000	152.000000	50%	2.000000	48.000000	3103.000000
75%	225.000000	39.000000	194.000000	75%	2.000000	61.000000	3656.250000
max	299.000000	54.000000	299.000000	max	3.000000	99.000000	4499.000000

Figure 1:Dataset Basic Statistics

Using .isnull().sum() function, the dataset appears to be well structured, with no missing values in any columns, which simplifies analysis and model-building by eliminating the need for imputation or handling null values. Each feature contributes fully to the dataset, covering a wide range of health, genetic, lifestyle, and demographic attributes. Notably, columns like *Insulin Levels*, *Blood Pressure*, *Cholesterol Levels*, *Glucose Tolerance Test*, *Genetic Markers*, and *Family History* provide comprehensive insight into factors potentially affecting the *Target* variable. The dataset also includes detailed lifestyle factors (*Physical Activity*, *Dietary Habits*, *Smoking Status*, *Alcohol Consumption*), as well as medical history factors(*History of PCOS*, *Previous Gestational Diabetes*), which could help in understanding the potential predictors for health outcomes related to the target variable.

1.2 Data Visualization

The correlation matrix below shows the relationships between various health-related numerical features, with each cell representing the Pearson correlation coefficient between two attributes. . This matrix is useful for identifying potential predictors for health outcomes and understanding how different attributes interact with one another.For instance, *Age* has a strong positive correlation with *Blood Pressure* (0.76), *Cholesterol Levels* (0.73), and *Waist Circumference* (0.73), suggesting that these metrics tend to increase with age. *Blood Pressure* is also highly correlated with *Cholesterol Levels* (0.67), indicating a close association between these two factors. Conversely, *Blood Glucose Levels* show weaker correlations with most other variables, suggesting they might be less directly influenced by other measured features in this dataset.

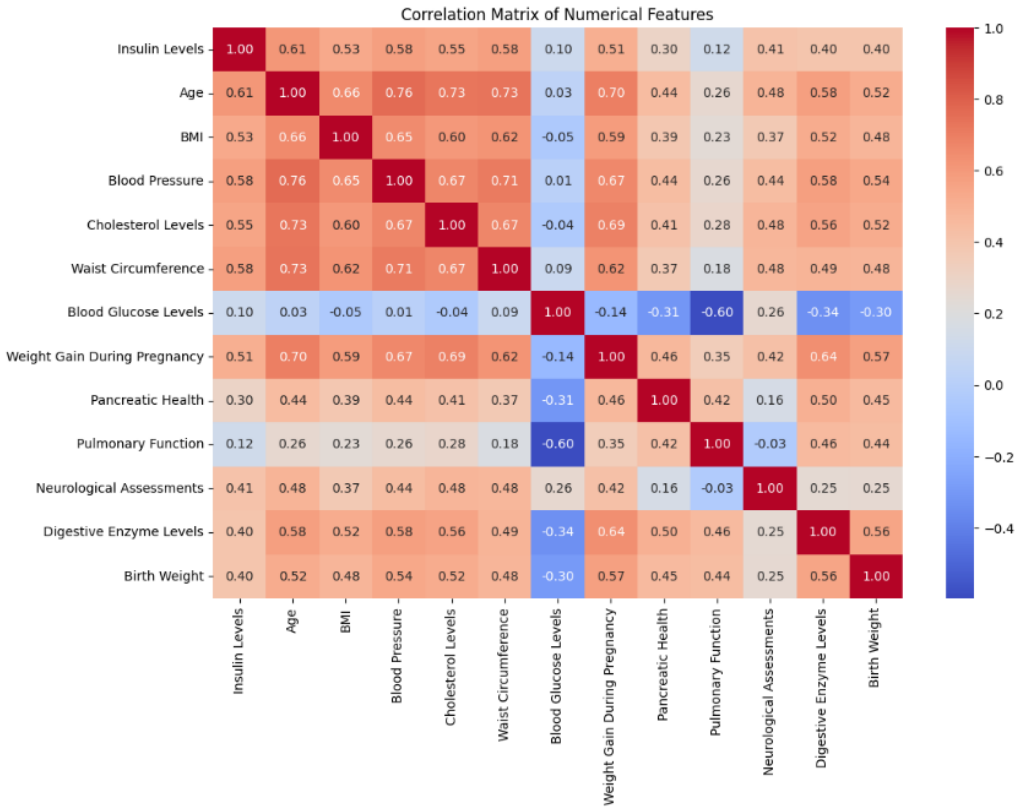

Figure 2:Correlation matrix for numerical features

In figure 3, the histogram shows the age distribution of various forms of diabetes and related conditions. For example, neonatal diabetes is nearly exclusively observed in neonates, due to its early beginning. However, adults, particularly those in middle age and beyond, are far more likely to have prediabetes, Type 2 diabetes, and LADA. Although they affect people of all ages, diseases like Type 1 diabetes are more prevalent in kids and teens. It is evident that several uncommon diseases, such as Wolfram syndrome and diabetes associated with cystic fibrosis, can manifest earlier in life. This breakdown shows how different diabetes issues appear at different phases of life and emphasizes the substantial correlation between age and the chance of developing forms of the disease.

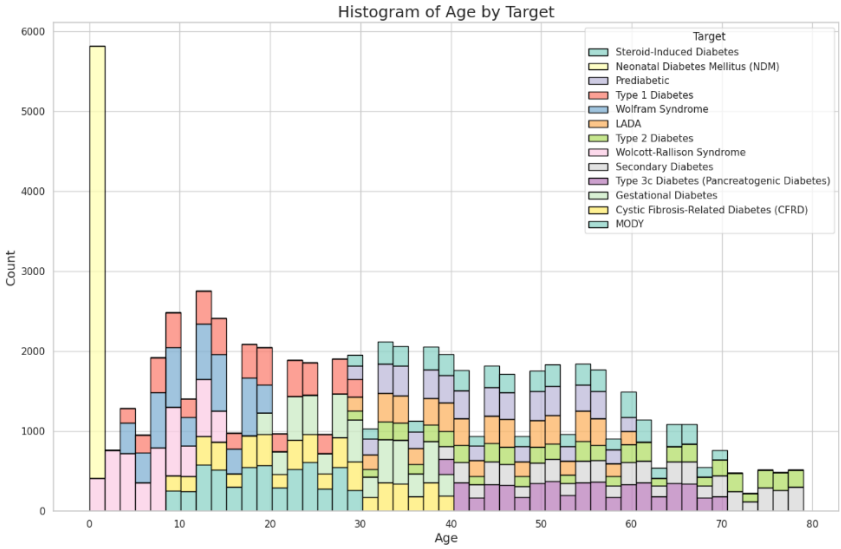

Figure 3: the age distribution of various forms of diabetes Histogram

The histograms Below display the distribution of various numerical features in the dataset. Many roughly normal distributions, though some are slightly skewed. *Age* and *Blood Pressure* display a moderate right skew, suggesting a higher frequency of younger ages and lower blood pressures in the dataset. *Insulin Levels* and *Blood Glucose Levels* have more irregular distributions, indicating potential subgroups or varying health conditions among participants. *Neurological Assessments* display discrete values, suggesting they may represent categorical or ordinal scales despite being treated as numerical.

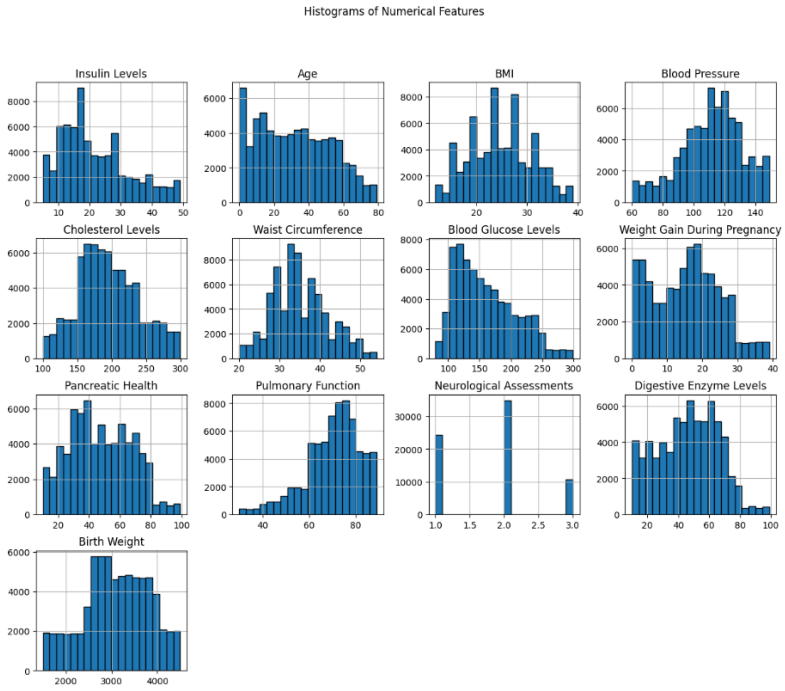

Figure 4: the distribution of various numerical features Histogram

The box plot below shows the distribution of cholesterol levels among the several target groups. With a wide interquartile range (IQR), we find that those with steroid-induced diabetes have the highest median cholesterol levels, suggesting significant variation within this population. Conversely, those with Type 1 diabetes and newborn diabetes mellitus (NDM) typically have lower median cholesterol levels, with NDM exhibiting very low variability. While gestational diabetes and cystic fibrosis-related diabetes (CFRD) exhibit comparatively lower medians and IQRs, conditions like Type 2 diabetes and Wolfram syndrome also exhibit elevated cholesterol levels.

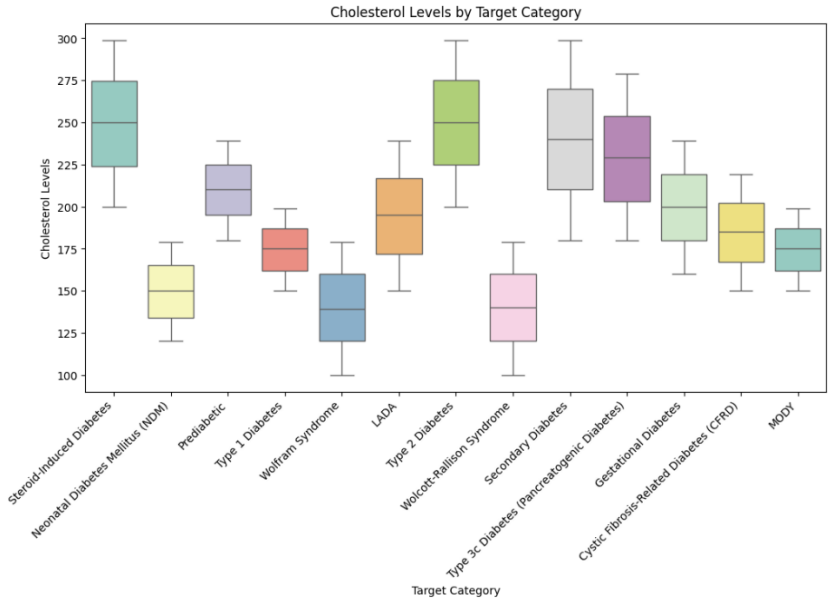

Figure 5: the distribution of cholesterol levels among the several target box plot

The scatter plot shows the association between blood pressure and age across several target groups. This distribution raises the possibility of age-related patterns in the prevalence and blood pressure traits linked to various forms of diabetes and associated illnesses. Each point, color-coded according to the goal category, symbolizes an individual. With most target categories showing up throughout a broad age range, the plot demonstrates that blood pressure typically rises with age. While Type 2 diabetes and Type 3c diabetes are common in older age groups, certain diseases, such as newborn diabetes mellitus (NDM), are only present in younger people.

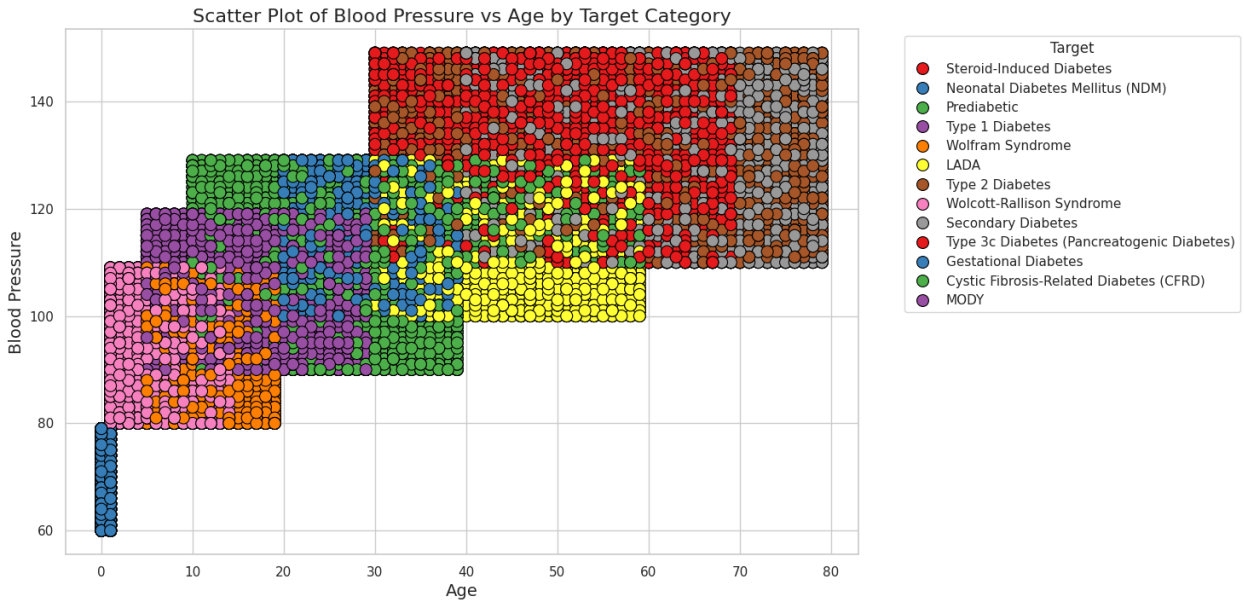

Figure 6: Scatter Plot of Blood Pressure vs Age by Target Category

1.3Data Cleaning

By plotting the box plots, to visually inspect for outliers in each numerical column of Diabetes Dataset, which can be important for understanding the spread and identifying extreme values.

Below figures show the outliers in Pulmonary Function and Waist Circumference columns, the others columns doesn't have outliers values.

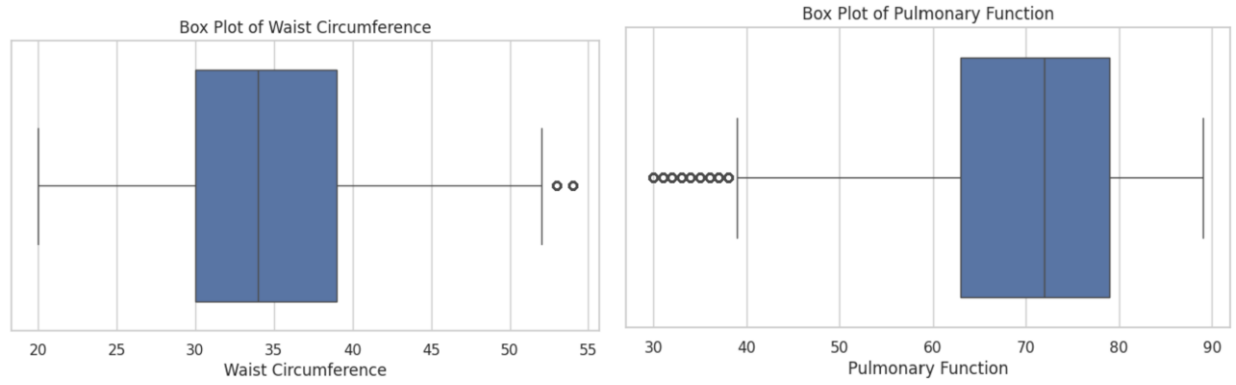

Figure 7:Box Plot Of Outliers

So, the code handles outliers in the Diabetes Dataset using the Interquartile Range (IQR) method, which identifies outliers based on the spread of the middle 50% of the data. For each numerical feature, by calculates the 25th percentile (Q1) and the 75th percentile (Q3), then computes the IQR as the difference between Q3 and Q1. Using this IQR, it defines lower and upper bounds for normal data as Q1 - 1.5 * IQR and Q3 + 1.5 * IQR, respectively. So the values outside these bounds are considered outliers, to handle them we can use the cap method, the outliers are capped or limited to the nearest bound. values below the lower bound are set to the lower bound, and values above the upper bound are set to the upper bound.

To ensures that extreme values do not skew the analysis we can compare the statistics before and after Capping method.in the below figures we can see that the statistics before and after haven't change much.

	Waist Circumference	Pulmonary Function
count	70000.000000	70000.000000
mean	35.044229	70.350271
std	6.783695	11.716294
min	20.000000	39.000000
25%	30.000000	63.000000
50%	34.000000	72.000000
75%	39.000000	79.000000
max	52.500000	89.000000

	Waist Circumference	Pulmonary Function
	70000.000000	70000.000000
	35.051657	70.264671
	6.803461	11.965600
	20.000000	30.000000
	30.000000	63.000000
	34.000000	72.000000
	39.000000	79.000000
	54.000000	89.000000

Figure 8:After Handle Outliers and before Handle Outliers

1.4Feature Engineering

1.4.1 Encode categorical variables into numerical formats

I applied the label encoding to all categorical columns in the DiabetesDataset. Wich it assigns a unique numerical value to each category within a column. So, diabetes_dataset_encoded was created by transforming the categorical data into numeric values, I was chosen label encoding over one-hot encoding because the categorical columns have a small number of unique categories. Also, the LabelEncoder stored the objects for each column in the label_encoders dictionary, allowing the user to reverse the encoding later if needed. So, the new DataFrame fully numeric, with all categorical columns replaced by their respective integer labels. In the figure below, we can see that the object converted to numeric value, for example, in Target column each unique value transferred to numeric value .

Target	Genetic Markers	Autoantibodies	Family History	Environmental Factors	Insulin Levels	Age	BMI	Physical Activity	Dietary Habits	...	Pulmonary Function	Cystic Fibrosis Diagnosis
7	1	0	0	1	40	44	38	0	0	...	76	0
4	1	0	0	1	13	1	17	0	0	...	60	1
5	1	1	1	1	27	36	24	0	1	...	80	1
8	0	1	0	1	8	7	16	1	1	...	89	1
12	0	0	1	1	17	10	17	0	0	...	41	0

Figure 9:Encoded Columns

1.4.2 normalize numerical features

I used Min-Max Scaling to normalize the numeric columns in the diabetes_dataset_encoded DataFrame. This scaling transforms the values of the numerical columns to a range between 0 and 1, which helps ensure that all features contribute equally to machine learning models. The MinMaxScaler is applied for all numeric columns except target column remain unchanged. The result is a new dataset, df_Normalized_Enco, where the numeric features are normalized, preventing features with larger ranges from dominating the learning process wich this improve model performance. In the figure below, we can see the normalized numeric value.

Liver Function Tests	Digestive Enzyme Levels	Urine Test	Birth Weight
1.0	0.516854	0.333333	0.376459
1.0	0.202247	0.000000	0.127042
0.0	0.505618	0.333333	0.707569
0.0	0.561798	0.333333	0.680894
1.0	0.157303	1.000000	0.090030

Figure 10:normalized columns

1.4.3 Analyze the relevance of each feature for the machine learning task.

To evaluate the relevance of numerical features for a machine learning task, I applied VarianceThreshold method, which it remove features with low variance, as they likely provide minimal information for prediction. A threshold of 0.045 is applied, meaning features with variance below this value are excluded. Out of the 13 original numerical features, 11 meet the threshold. This process ensures that the dataset includes only the most informative features, reducing noise and improving the efficiency of the machine learning model.The figure below shows the correlation matrix for the selected features

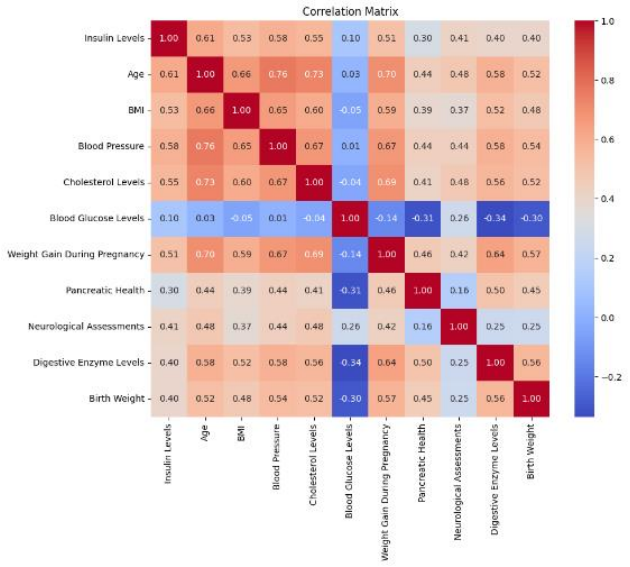

Figure 11:correlartion matrix

1.4.4 reduction techniques to reduce data dimensionality

Principal Component Analysis (PCA) was applied to the features selecteion dataset to reduce its dimensionality while preserving 90% of the variance. This means that PCA identifies the most important features and transforms the original data into a smaller number of new features, called principal components. The resulting components are stored in a new DataFrame(pca_df). The number of components chosen is 15, which are sufficient to capture the desired 90% of the variance from the original data. The target variable is added back to the dataset. This will help in reducing the complexity of the data while retaining most of the relevant information, which can improve the performance and interpretability of models.

1.5 Model Evaluation:

1.5.1 Split the dataset into training and testing subsets.

the dataset was splited into training and testing sets using train_test_split from scikit-learn. First, the raw features (X_raw) and target variable (y_raw) are separated from the diabetes_dataset_encoded, while the processed features (X_preprocessed) and target variable (y_preprocessed) come from the PCA-transformed dataset (pca_df). Then, the data is split into 75% training and 25% testing sets for both the raw and processed datasets using a fixed random seed (random_state=42).

1.5.2 Train on raw data & preprocsessed Data

A Random Forest Classifier is trained on both the raw data (X_train_raw) and the processed data (X_train_preprocessed) to evaluate and compare their performance. The classifier is trained on each dataset and then used to predict the target variable (y_pred_raw for raw data, y_pred_preprocessed for processed data).

1.6 Experiments and results

1.6.1 Compare the results of a Random Forest model trained on the preprocessed data versus the raw data.

The preprocessed data (after PCA) trained model shows an exceptionally high performance, with accuracy, precision, and recall all close to 1.0. This suggests that dimensionality reduction through PCA has significantly improved the model's ability to make accurate predictions.

```
Performance on Processed Data:
Accuracy: 0.9999
Precision: 0.9999
Recall: 0.9999
```

Figure 12:performance on preprocessed data

On the other hand, the model trained on the **raw data** performs considerably worse, with accuracy, precision, and recall ranging around 0.90. This indicates that the model might be struggling with irrelevant or less informative features, which could cause overfitting or underfitting. The raw dataset likely contains redundant or noisy features, which can dilute the model's ability to focus on the most important patterns in the data.

1.6.2 Analyze the effect of various preprocessing techniques on model performance using metrics such as accuracy, precision, and recall.

This part helps to understanding how different transformations can impact a machine learning model's ability to make accurate predictions. Key preprocessing steps such as scaling, dimensionality reduction (PCA), feature selection, and label encoding all play a role in improving model performance. Scaling normalizes the range of features, which is especially beneficial for distance-based or gradient-based models but has a minimal impact on tree-based models like Random Forests. Label encoding is useful for convert the categorical data. Dimensionality reduction through PCA and feature selection techniques like

variance thresholding help reduce noise and overfitting, which enhances the model's ability to generalize to new data.

So, preprocessing techniques such as PCA and feature selection can simplify and improve the data representation, making machine learning models more effective. For Random Forest models, scaling had minimal impact, but PCA and feature selection led to significant improvements in accuracy, precision, and recall by reducing noise and focusing on key features. While label encoding was effective for ordinal data, its impact depends on the nature of the categorical features. By carefully applying these preprocessing steps, model performance can be significantly enhanced, as evidenced by the improved metrics when using processed data compared to raw data.

1.6.3 Results "Summarize the performance of the model and improvements in model accuracy, consistency, and training speed due to preprocessing"

The model demonstrated significant improvements in performance when trained on preprocessed data compared to raw data. With preprocessing techniques like PCA and feature selection, the model achieved an accuracy of 0.9999, precision of 0.9999, and recall of 0.9999, compared to the raw data's accuracy of 0.9012, precision of 0.9061, and recall of 0.9012. These improvements shows how preprocessing reduced noise, eliminated irrelevant features, and preserved key variance, leading to better generalization and model effectiveness. Additionally, the reduced dimensionality from PCA likely sped up training by focusing the model on the most important features. Finally, the model after applying PCA shows a significant performance boost, with improvements around 9–10% in accuracy, precision, and recall.

Part 2

2.1 Model Training:

The dataset is split into training and testing sets (75% for training, 25% for testing).then evaluate the models based on four key performance metrics: accuracy, precision, recall, and F1-score, also tracks the training time for each model. Random Forest, SVM, and MLP are trained and evaluated using the train_and_evaluate_[model Name] function, which measures the time taken to train the model, makes predictions, and calculates performance metrics. Then the results are showing the evaluation metrics for each model, including the training time required for each one which provides a comprehensive comparison of the models' effectiveness and computational efficiency.

2.2 Models Results and Comparison:

2.2.1Performance Analysis:

The Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) models performed exceptionally well on the dataset. Random Forest achieved near-perfect results with an accuracy, precision, recall, and F1-score of 0.9999 where SVM and MLP delivered perfect metrics across the board, with values of 1.0000 for all performance measures. While all three models demonstrated outstanding classification capabilities, SVM achieved this with the lowest training time, and MLP followed it with nearly close but RF,

despite its excellent performance, took significantly longer to train, reflecting its higher computational demands

```
Comparison of Classification Models:
Random Forest Performance:
Accuracy: 0.9999, Precision: 0.9999, Recall: 0.9999, F1-Score: 0.9999, Training Time: 52.9377 seconds

SVM Performance:
Accuracy: 1.0000, Precision: 1.0000, Recall: 1.0000, F1-Score: 1.0000, Training Time: 12.7542 seconds

MLP Performance:
Accuracy: 1.0000, Precision: 1.0000, Recall: 1.0000, F1-Score: 1.0000, Training Time: 19.7886 seconds
```

Figure 13:Models results

2.2.2Strengths and Weaknesses

Random Forest is robust to noise and doesn't require extensive preprocessing, making it ideal for diverse datasets. However, it is slower to train. SVM excels with smaller, well-separated datasets, achieving high accuracy with low computational cost, but it can be sensitive to hyperparameters like the kernel and C. MLP is highly flexible and capable of modeling complex, non-linear relationships, making it suitable for intricate datasets. However, MLP requires careful parameter tuning, has longer training times than SVM, and may consume more memory due to its iterative optimization process.

2.2.3 Comparison: Computational Efficiency

SVM is the most computationally efficient, completing training in just 12.75 seconds, followed by MLP at 19.79 seconds. RF, while powerful, took the longest time at 52.94 seconds due to its ensemble nature. SVM's lightweight computation and simplicity make it an attractive choice for quick and efficient training.

So, for this dataset, SVM provides the best balance between prediction accuracy and computational efficiency, making it the most suitable choice when training time is critical. MLP offers a flexible alternative, excelling with complex datasets at a moderate computational cost. Random Forest remains a strong contender, particularly for datasets with noise or when feature importance interpretation is necessary but its slower training time makes it less practical for time sensitive tasks.

2.3.4 Effect of Preprocessing on Model Performance

We can see that preprocessing techniques that were be used had a transformative impact on model performance, significantly enhancing accuracy, precision, recall, and F1-scores while improving computational efficiency. So, before preprocessing, Random Forest (RF) outperformed Support Vector Machine (SVM) and Multilayer Perceptron (MLP) in accuracy (0.9009 vs. 0.7518 and 0.8282, respectively), but after preprocessing, all models achieved near perfect or perfect metrics, with SVM and MLP showing the most substantial gains. SVM reduced its training time from 142.62 seconds to just 12.75 seconds, and MLP improved accuracy from 0.8282 to 1.0000 while halving its training time. RF maintained its performance, though with increased computational demands due to the complexity of transformed data. These results underscore the vital role of preprocessing in optimizing machine learning model accuracy and efficiency.

```
Random Forest Performance:
Accuracy: 0.9009, Precision: 0.9058, Recall: 0.9009, F1-Score: 0.8998, Training Time: 16.5473 seconds

SVM Performance:
Accuracy: 0.7518, Precision: 0.7522, Recall: 0.7518, F1-Score: 0.7517, Training Time: 142.6219 seconds

MLP Performance:
Accuracy: 0.8282, Precision: 0.8285, Recall: 0.8282, F1-Score: 0.8272, Training Time: 237.0337 seconds
```

Figure 14:raw data results

```
Comparison of Classification Models:
Random Forest Performance:
Accuracy: 0.9999, Precision: 0.9999, Recall: 0.9999, F1-Score: 0.9999, Training Time: 52.9377 seconds

SVM Performance:
Accuracy: 1.0000, Precision: 1.0000, Recall: 1.0000, F1-Score: 1.0000, Training Time: 12.7542 seconds

MLP Performance:
Accuracy: 1.0000, Precision: 1.0000, Recall: 1.0000, F1-Score: 1.0000, Training Time: 19.7886 seconds
```

Figure 15:preprocessd data

2.3.5 Effect of Model Parameters: Investigate how parameters affect model performance. Try different combinations of parameters.

Random Forest Classifier

I used GridSearchCV to optimize the hyperparameters of a Random Forest Classifier to improve its performance. The parameter grid included options for the number of trees (n_estimators), the maximum depth of the trees (max_depth), the minimum samples required to split a node (min_samples_split), and the minimum samples required at a leaf node (min_samples_leaf). By performing a 3-fold cross-validation, the grid search tested 16 different combinations of these parameters. The best parameters were found to be n_estimators=100, max_depth=None, min_samples_split=2, and min_samples_leaf=1, which resulted in a cross-validation accuracy of **0.9998**. This shows that tuning the model's parameters can significantly enhance its ability to make accurate predictions.

Support Vector Machine

I also used GridSearchCV to optimize the hyperparameter C for a Support Vector Machine **(SVM)** classifier. The grid included three values for C (0.1, 1, and 10), which controls the trade-off between achieving a low error on the training data and maintaining a low margin for misclassifications. Using 3-fold cross-validation, the grid search evaluated each combination, totaling 9 fits. The best parameter was determined to be C=0.1, achieving a perfect cross-validation accuracy of **1.0000**. This demonstrates that tuning even a single hyperparameter can result in significant improvements in the model's performance.

Multilayer Perceptron (MLP)

I applied GridSearchCV to optimize the hyperparameters of a Multilayer Perceptron (MLP) **classifier**. The grid included combinations of hidden_layer_sizes (e.g., (50,), (100,), (50, 50)), learning_rate strategies (constant and adaptive), and initial learning rates (learning_rate_init

values of 0.001 and 0.01). Using 3-fold cross-validation, the search evaluated 12 combinations across 36 fits in total. The best configuration was determined to be hidden_layer_sizes: (50,), learning_rate: 'constant', and learning_rate_init: 0.001, which achieved a perfect cross-validation accuracy of **1.0000**. This optimization shows the importance of tuning multiple hyperparameters to maximize the performance of neural network models.

Conclusion

In conclusion, this assignment highlights the important role of data preprocessing in improving the performance of machine learning models. By applying data cleaning and feature engineering techniques, we saw better results with the Random Forest model, showing how crucial proper data preparation is. The comparison between Random Forest (RF), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) models revealed that each has its own strengths, but the best choice of model depends on the specific characteristics of the dataset and available computational resources. Tuning the model's parameters and preprocessing the data also had a significant impact on performance, emphasizing the need for a careful approach when selecting and optimizing models. In the end, our analysis helped identify the model that offers the best balance between prediction accuracy and computational efficiency for predicting bike demand, providing useful insights for future work in predictive analytics and intelligent systems.